Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 37 (2014) 244 – 251
Procedia Computer Science

The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2014)

Classification of Post-Deployment Performance Diagnostic Techniques for Large-Scale Software Systems

Haroon Malik[a]*, Elhadi M. Shakshuki[b]

[a]*David R. Cheriton School of Computing, University of Waterloo, Waterloo , ON, N2L3G1, Canada*
[b]*Jodery School of Computing, Acadia University, Wolfvilee, NS, B4P2R6, Canada*

Abstract

Today's large-scale software systems (LSSs) such as Facebook, Google, Amazon and many other contemporary datacenters comprise hundreds or thousands of machines running complex applications that require high availability and responsiveness. These LSSs must be carefully monitored for performance bottlenecks before a serious harm is done. Performance analysts have to deal with the tedious task of monitoring the performance of these LSSs to avoid any service level agreements (SLA) violations and to ensure their failure free operations. There do exist several post-deployment performance diagnostic (PPD) techniques for to help analysts diagnose performance problems in the field, i.e., after the software is deployed. However, there is no classification of the proposed PPD techniques to understand their objectives and characteristics. In this paper, we classify the existing PPD techniques along multiple categories. The classification of PPD techniques will provide a guideline for performance analysts and practitioners of LSS to choose techniques suitable for their need. Moreover, the classification will also help researcher understand and fill gaps, i.e., dedicate their research efforts to categories that have received little attention in the past.

Peer-review under responsibility of the Program Chairs of EUSPN-2014 and ICTH 2014.

Keywords: Large-Scale Software System; Performance Monitoring; Performance Diagnostics; Anomaly Detection;

1. Introduction

Today's large-scale systems (LSS) consisting of datacenters and server farms have experienced an extraordinary explosion in size and complexity. For example, Google the fifth largest data center alone maintains a pool of more than one million servers [1]. Facebook has doubled the size of its data center within a year from 60,000 to 120,000

* Corresponding author. Tel.: +1-519-957-2262;
E-mail address: malikh@uwaterloo.ca

Peer-review under responsibility of the Program Chairs of EUSPN-2014 and ICTH 2014.
doi:10.1016/j.procs.2014.08.036

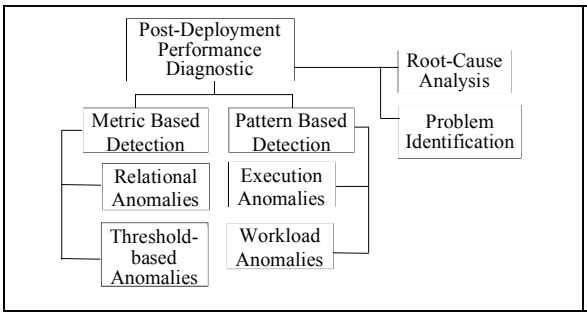

Fig.1. Taxonomy for PPD techniques for large-scale software systems

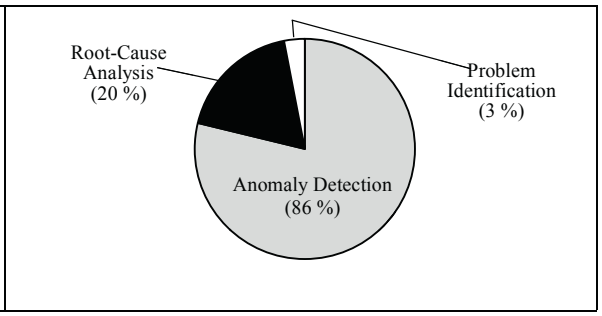

Fig. 2. Composition of PPD work

servers and continues to grow [2, 3]. Similarly, the collective server count of Microsoft, eBay, Yahoo and Amazon is over 550,000 servers [4-6].

LSSs are large capital investments for the service providers. Any discrepancy in their performance can result in large monetary losses. For example, an hour-long PayPal outage may have prevented up to $7.2 million in customer transactions [7]. Therefore, operators of LSS closely monitor their LSS to anticipate and identify performance bugs before a violation of service level agreements (SLAs) and unplanned system downtime that can cost as much as $550,000 per hour in lost revenues of LSS [8].

A performance problem includes an application or a system not responding fast enough, taking too much of an important resource, crashing or hanging under heavy load, or not meeting the desired service-level agreements (SLAs) [9]. Performance problems with LSS typically exhibit symptoms such as high response time, increased latency and low throughput under load.

The occurrence of performance problems in ever-growing LSSs has become a norm, rather than an exception [10]. Performance problems after deployment/expansion (i.e., post-deployment) of LSS are seldom due to feature errors, rather due to systems not scaling to field workloads. LSSs like AT&T and BlackBerry also report their concerns about performance degradation and resource saturation as the fundamental post-deployment problems [11]. For example, a robust performance monitoring and diagnostics could have alerted the operators of Skype for the system overload that resulted in system availability disruption for 48 hours, leaving millions of users without service [12].

Although many techniques have been introduced in the literature to diagnose post-deployment performance problems (e.g., [3,4,5,6,8,9,10,11,12,13,14,15]), there is no classification to understand the objectives and characteristics of these techniques. Moreover, non-existence of a classification provides a) little or no direction to practitioners in choosing the appropriate PPD techniques to satisfy their respective performance diagnostic objective and b) for researcher, identify gaps, i.e., dedicate their efforts to the classification category, where little work is done.

In this paper, we survey the state-of-the-art PPD techniques aimed to diagnose performance problems of LSS in the field, i.e., post-deployment performance problem. We first classify the PPD techniques into three major categories, i.e., anomaly detection, root/cause analysis and problem identification. We further categories anomaly detection category into two major sub-categories based on detection techniques, i.e., a) metric based detection techniques that capture both relational and threshold-based anomalies and b) pattern detection techniques that are geared to diagnose both execution and workload anomalies as shown in Fig.1. The survey will help researchers to differentiate existing PPD techniques. Moreover, it will provide a guideline to practitioners to choose an appropriate PPD technique suitable for their needs.

2. Performance Diagnostic

LSS contains multiple subsystems that interact across multiple nodes in sometimes unforeseen and complicated ways. As a result, detecting, isolating and identifying the root-cause of a performance problem is frustrating and can

take several hours or even days. We donate such activity as *performance diagnostics*. Based on surveyed papers, we classify the performance diagnostic techniques into two major categories, i.e., post and pre-deployment. By *pre-deployment diagnostics*, we refer to the spectrum of performance testing (regression, load and stress). In the course of performance testing, load on a system is placed (within or above its design constraints) to test whether a system is able to support a specific demand that resembles the field usage intensity. The works that falls under pre-deployment diagnostic category helps analyst to identify isolate and pinpoint performance failures before they become critical field problems. Such performance problems include performance bottlenecks, performance bugs (i.e., system freezes, crashes and becoming unresponsive during the course of the test) and early design problem. Whereas, by *post-deployment diagnostics*, we refer to the work that are dedicated to identifying performance problems in the field. Examples of such problems are response time degradation, higher than expected resource utilization, or systems not scaling to field workloads. We detail our classification of PPD techniques shown in fig. 1 in the subsequent subsections.

2.1. Anomaly Detection

The objective of the works included in the anomaly detection category is to detect performance deviation(s) of a system with that of its expected behavior by the use of models, thresholds and SLAs. The anomaly detection based techniques perceives the system behavior (e.g., current disk utilization) that cannot be explained by the observed workload (e.g., the type and volume of transaction processed). From the surveyed papers undertaken in our study, we classify anomaly detection techniques for PPD into two major types, i.e., metric and pattern based anomaly detection techniques. We further classify the metric and sequence based anomaly detection techniques into their respective subcategories based on the intuition of researchers to diagnose a specific type of performance problems.

2.1.1. Metric-based anomaly detection techniques

Metric-based anomaly detection techniques make use of one or more performance metrics (usually numerical) and compare them with a baseline metric value(s) to detect anomalous transactions, component or system behavior. For example, several researchers have developed different types of metric based anomaly detection techniques to identify performance problems when a resource, such as CPU or disk utilization or derived metrics, such as throughput or latency goes beyond a desired/agreed threshold values. From the surveyed papers, we are able to classify two subcategories for metric based anomaly detection techniques. Furthermore, we explain each subcategory. The two sub-categories of metric based performance anomalies are:

2.1.1.1. Relational anomalies

The works categorized in this category characterize the normal behavior of the system by building relationship(s) between performance metrics, e.g. performance counters, system metrics and system invariants [6-9]. The normal behavior of the system entails finding and modeling stable relationships between the performance metrics. For example, for a given system load, if 80% of the CPU is consumed, then the disk utilization is always around 20%. The relationships are mathematically modeled and tracked for checking system health. A broken relationship between performance metrics is flagged as relational anomaly. In similar capacity, the anomalous behavior of the system is flagged based by observing how the monitoring data reacts to the volume of user requests (merged). For example, in large transaction processing systems, flagging of relational anomaly is made possible by tagging and monitoring these users request as they flow through a different component/subsystems of LSS, i.e., web servers, databases, and application servers.

2.1.1.2. Threshold-based anomalies

The main objective behind threshold-based anomaly detection work is to compare the performance metric value against its acceptable limits, i.e., thresholds. For example, if a selected metric (e.g. CPU usage) exceeds a given threshold, an alert is generated to notify operator of LSS for a threshold-based performance anomaly, who might follow-up with an examination [17]. Similarly, anomalous node(s) in LSS are identified if the confidence level reported by statistical technique(s) on the node's performance metrics falls below an estimated threshold value [18].

2.1.2. Pattern-based anomaly detection techniques

Pattern-based anomaly detection techniques extract and model the pattern. The pattern can be a) recurring set of sequences, such as event sequences in performance logs or data sequences (such as fixed-length sequence or variable-length sequence) in data streams or b) objects, such as clusters and graph trajectories derived from the numerical data of performance metrics. Mostly, machine learning techniques are used to determine the abnormality of the sequences, i.e., difference in the sequences of events or difference in the sequence of elements within an event. Mostly, pattern recognition techniques are used to determine the abnormality in the object's structure. Many researchers have developed pattern based anomaly detection techniques that model the amount of transaction/traffic handled by a component under various normal conditions. The pattern based anomaly detection techniques, alert operator for a possible anomaly when a different pattern of transaction/traffic flow is observed. Techniques meant for detecting metric based performance anomalies cannot detect the pattern based anomalies such as anomalous behavior of the system arising due to abnormal flow of transaction/traffic to a system and vice versa. The two sub-categories of metric based performance anomalies are workload anomalies and execution anomalies

2.1.2.1. Workload anomalies

The objective behind workload anomaly detection is to characterize the normal workload induced in the system. Anomalies based on unexpected background or interfering workloads such as unscheduled virus scan or an unannounced replication between servers are pointed out (merged). For example, Sandeep et al. [19] used a statistical technique (principal feature analysis (PFA)) to characterize the workload processed by a system and then a machine learning technique (random forest) to infer workload anomalies.

Sandeep used PFA [21] technique to reduce the performance counter set collected under the normal system behavior for a given workload. In the reduced counter set, some counters are more important to characterize the workload than others. Thus, Sandeep further used a standard machine learning technique called random forest [22], which uses decision trees to classify labeled data. The output of random forest supervised learning algorithm produced a ranking of counters by their importance in classifying the observations. Given the ranked list of counters, Sandeep figured out the cutoff point beyond which the information value of the counters is low by the use of Clara clustering [23]. By clustering on an increasing subset of the ranked counters, Sandeep discovered the smallest set of counters that distinguishes the workload more effectively and called it workload signature profile. The signature of the given workload is the medoid of its cluster. Workload anomaly is detected by calculating the proximity between the test data and the medoids.

Similarly, Hoke et al. (merged) used a statistical technique to identify workload anomalies based on stream data arising from sensors. Hoke used a variant of the principal component analysis technique called spirit [26]. The stream data, i.e., number of packets sent over the network across n-hosts are grouped into an n-dimensional vector and then spirit algorithm is applied to project this vector into a k-dimensional space. The projections are hidden variables. If the number of dimensions in this space changes, this signifies a workload anomaly.

Many researchers have defined techniques to infer the workload anomalies based on background workload caused by conflicting policies in self-tuning and performance-sensitive systems. For example, Heo et al. [27] used a mechanism called adaptation graph analysis for identifying background workload triggered by two conflicting policies, i.e., On/Off and dynamic voltage scaling (DVS) policies (merged). Heo presented a case study of bad

interactions between two energy saving policies leading to an increase in the energy consumption in a multi-tier web server farm.

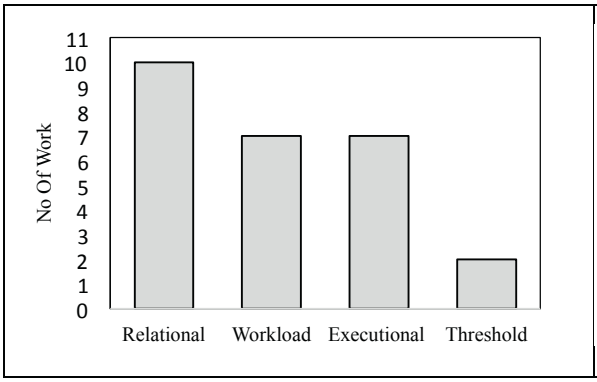

Fig. 3. Distribution of Performance Anomaly Detection Work

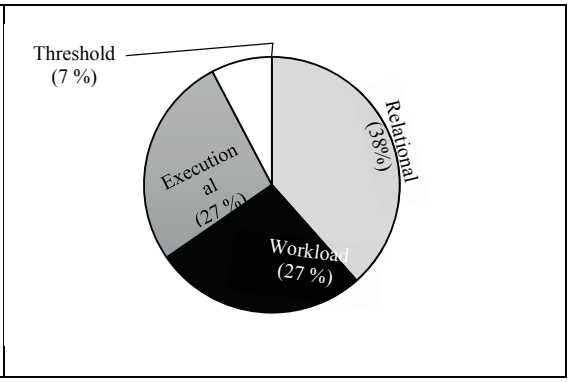

Fig. 4. Composition of Performance Anomaly Detection Work

When the system was underutilized, a DVS policy reduced the CPU frequency to save energy. This caused an increase in machine utilization and delay, which in turn triggered an allocation policy to turn on an extra machine. Once the load was balanced across a larger set of machines, the cycle repeated itself with the DVS policy, reducing CPU frequency further, ultimately waking up more machines than needed, and thereby wasting energy. Similarly, Khan et al. [30] developed a technique by extending the discriminative pattern analysis (a data mining technique) to recognize cyclic event pattern arising due to conflicting components that accelerate performance degradation over time.

2.1.2.2. Execution anomalies

Execution anomalies include both 1) workflow errors and 2) execution low performance problems. Workflow errors occur along the software execution paths. More precisely, workflow error refers to event(s) that do not synchronize with pre-defined sequences in a software. In execution low performance problems, the execution time of a job takes much longer than normal case/scenario, although its execution path is correct [31]. Console logs produced by system have been used to detect execution anomalies. The idea is simple. When log messages are grouped properly into message sequences, there is a strong and stable correlation between messages within the same group. This is because they are likely to come from logically related executions steps in the program. An unusual sequence of a message group marks an execution anomaly.

2.2. Analysis of anomaly detection work

Fig. 2 shows the composition of the PPD works. Among the thirty PPD surveyed works, (26)86% of them are about performance anomaly detection. Among these 26 surveyed works, (12) 46% of them present metric bases anomaly detection techniques. Whereas, remaining (14) 54% of the surveyed works have proposed sequences bases anomaly detection techniques. Fig. 4 reveals that a major portion of the post-deployment anomaly detection works, i.e., 10 (38%) focus on identifying relational anomalies followed by 7 (27%) of work that aims to identify execution anomalies. There are 2 (7%) works dedicated for pinpointing anomalies based on threshold i.e., threshold-based anomalies. The focus of 7 (27%) works in anomaly detection category is to identify workload anomalies. The sum (%) is more than 100%, since many of the works have presented techniques that cover the detection of multiple catagories of performance anomalies. Fig. 3 illustrates the distribution of performance anomaly work across thirty surveyed papers. An unexpected observation about the anomaly detection work is that there is a little use made of stream data to detect performance anomalies, i.e., only 2 (5%) of the anomaly detection works. Despite the fact, that emerging LSSs are awash in monitoring data collected collected from infrastructure components such as chillers,

UPSes, power distribution units and wiresless sensors. These infrastructure components produce large streams of performance data.

Table 1. Summary of Post-deployment Performance Diagnostic Work

Work	Published Date	Reference	Anomaly Detection	Problem Identification	Root-Cause Analysis	Work	Published Date	Reference	Anomaly Detection	Problem Identification	Root-Cause Analysis
Gunter et al.	(2000)	[13]	R	–	–	Xu et al.	(2009)	[35]	E	–	–
Jiang et al.	(2008)	[6]	R	–	–	Bodik et al.	(2009)	[38]	–	✓	–
Chen et al.	(2002)	[14]	R	–	✓	Cherkasova et al.	(2009)	[20]	W	–	–
Tierney et al.	(2003)	[12]	R	–	–	Jiang et al.	(2009)	[40]	R	–	✓
Cohen et al.	(2004)	[41]	–	–	✓	Fu et al.	(2009)	[16]	E	–	–
Kiciman et al.	(2005)	[15]	R	–	✓	Jiang et al.	(2009)	[17]	T	–	✓
Guo et al.	(2006)	[10]	R	–	–	Gunasekaran	(2010)	[39]	–	–	✓
Jiang et al.	(2006)	[11]	R	–	–	Xu et al.	(2010)	[32]	E	–	–
Hoke et al.	(2006)	[24]	W	–	–	Kutare et al.	(2010)	[36]	E, W	–	–
Hoke et al.	(2006)	[25]	W	–	–	Khan et al.	(2011)	[30]	W	–	✓
Munawar et al.	(2007)	[9]	R	–	–	Munawar et al.	(2008)	[8]	R	–	–
Sandeep et al.	(2007)	[19]	W	–	✓	Jiang et al.	(2008)	[31]	E	–	–
Peret et al.	(2007)	[18]	T	–	✓	Xu et al.	(2009)	[33]	E	–	–
Heo et al.	(2007)	[27]	W	–	–	Xu et al.	(2008)	[37]	E	–	–
Xu et al.	(2008)	[34]	E	–	–	Chandra et al.	(2008)	[7]	–	–	–

Legend: **R**— Relational, **W**—Workload, **T**— Threshold, **E** — Executional

2.3. Problem identification

The objective of works in this category is to help performance analysts quickly determine if the detected performance problem is similar to a previously seen performance problem. Hence, a known remedy to the identified performance can be applied; avoiding escalation and speeding up performance diagnostic process. One way to identify performance problems in LSSs such as data centers is to construct a datacenter fingerprint (fingerprints of the previously seen problems in performance repositories). The fingerprinting involves capturing and concisely summarizing the subset of the collected metrics that best discriminate among different performance crises [38].

2.4. Root-cause analysis

The objective of root cause analysis techniques is to characterize the exact cause of an identified problem so that the corrective action can be taken to ensure that the identified problem do not manifest again. For example, abnormal disk utilization identified as disk saturation may have been caused by background activities such as disk scrubs, RAID reconstructions, unexpected antivirus scan or an unannounced data replication. Once an anomaly is detected or performance problem is identified, the analyst determines the root cause by analyzing and interpreting large number of log messages. By intelligent grouping and correlating events in the log provide system administrators with a meaningful information in a concise format for root cause analysis [39]. Many monitoring techniques rank the faulty components (merged), deviated metrics (merged) and performance alerts [17] based on criteria

3. Implication for Research and Practice

Table 1 summarizes the proposed classification of PPD techniques. We list the implication of our propose classification for both research and practice:

1. Performance monitoring and diagnostics of LSSs and data centers is mostly objective centric, i.e., the performance of the system is perceived based on the customer demands, stakeholder concerns and business constraints. Our study reveals that researches have not done justice in providing crisp and clear performance objective(s) of their proposed PPD techniques. Our paper provides a taxonomy to facilitate researchers implicitly and correctly label their proposed PPD techniques.
2. We have used our taxonomy to classify the existing PPD research work. In this act, we have identified several areas that seem to have gotten little attention from the research community such as post-problem identification. Researchers now can put in their innovations to develop new tools/techniques in these neglected areas.
3. The most prominent implication of our study for the industry is that it allows practitioners to easily pick up PPD techniques based on their needs and performance objectives.

4. Conclusion

Research on large industrial projects has shown that the performance problems observed in the field are often performance related. There exists much research effort to help analysts and practitioner of these LSSs diagnose performance problems in the field and avoid monetary losses. However, there do not exist any effort to classify the PPD techniques based on their objective (i.e., type of performance problem and characteristics). We classified PPD techniques along three major categories, i.e., anomaly detection, root-cause analysis and problem identification. We further subcategories anomaly detection techniques into two major categories, i.e., a) metric based detection techniques that capture both relational and threshold-based anomalies and b) pattern-based anomaly detection techniques that are geared to diagnose both execution and workload anomalies. Analysis of the surveyed work indicates that that most of the research effort is dedicated towards helping practitioner detect performance anomalies in the field. However, there is little work done on problem identification, i.e., categorize, label, and pinpoint performance problems in the field, if it is similar to previously seen problem, so a quick fix can be applied. Researches need to dedicate their effort in developing innovating techniques that can help analysts in the automatic identification of performance problem in the field.

References

1. Gabriel R, Kazman R, Northrop L, Schmidt D, Sullivan K. Workshop on software technologies for ultra-large scale systems. Software engineering-companion. ICSE; 2007.
2. Facebook will double size of oregon data center [Internet]: Data Center Knowledge; 2010 [updated Jul; cited Feburary 2013]. Available from: http://www.datacenterknowledge.com/archives/2010/07/30/facebook-will-double-size-of-oregon-data-center/.
3. From 60,000 servers to 120,000! [Internet]; 2010.
4. Report: EC2 running 40,000 servers [Internet]; 2009.
5. Who has the most web servers? [Internet]; 2009.
6. Microsoft: 300,000 server in container farm [Internet]; 2008.
7. Lin LH, Tanyavutti A, Jindrapacha S. Analyzing eBay platform strategies: An application of meyer's product platform strategy model. Management of engineering and technology; 2007. p. 125-142.
8. Roblee C, Berk V, Cybenko G. Large-scale autonomic server monitoring using process query systems. Proceedings of the second IEEE international conference on autonomic computing (ICAC-05); 2005.
9. Jiang ZM, Hassan AE, Hamann G, Flora P. Automatic identification of load testing problems. IEEE international conference on Software maintenance; 2008. p. 44-2008.
10. LU X. Autonomic failure prediction based on manifold learning for large-scale distributed systems. 02. 02-2010;17(4):116-24.
11. Thakkar D, Hassan AE, Hamann G, Flora P. A framework for measurement based performance modeling. WOSP '08: Proceedings of the 7th international workshop on software and performance; Princeton, NJ, USA; 2008.
12. Restarts cited in skype failure [Internet]; 2007.
13. Jiang M, Munawar MA, Reidemeister T, Ward PAS. Information-theoretic modeling for tracking the health of complex software systems. conference of the center for advanced studies on collaborative research: Meeting of minds; Toronto, Canada. ACM; 15-2008.
14. Chandra A, Prinja R, Jain S, Zhang ZL. Co-designing the failure analysis and monitoring of large-scale systems. 14. 14-2008;36(2):10-5.

15. Munawar MA, Jiang M, Ward PAS. Monitoring multi-tier clustered systems with invariant metric relationships. Proceedings of the 2008 international workshop on software engineering for adaptive and self-managing systems; ACM; 16-2008.
16. Munawar M, Ward P. Leveraging many simple statistical models to adaptively monitor software systems. 17. 17-2007;4742/2007:457-70.
17. Fu Q, Lou JG, Wang Y, Li J. Execution anomaly detection in distributed systems through unstructured log analysis. 2009 ninth IEEE international conference on data mining; IEEE; 37-2009.
18. Kiciman E, Fox A. Detecting application-level failures in component-based internet services. 75. 2005;16(5):1027-41.
19. Guo Z, Jiang G, Chen H, Yoshihira K. Tracking probabilistic correlation of monitoring data for fault detection in complex systems. International conference on dependable systems and networks, 2006. DSN 2006; Philadelphia, PA. ; 20-2006.
20. Jiang G, Chen H, Yoshihira K. Discovering likely invariants of distributed transaction systems for autonomic system management. 21. 21-2006;9(4):385-99.
21. Tierney B, Gunter D. NetLogger: A toolkit for distributed system performance tuning and debugging. Proceedings of the 8th IFIP/IEEE international symposium on integrated network management; Citeseer; 25-2003.
22. Gunter D, Tierney B, Crowley B, Holding M, Lee J. Netlogger: A toolkit for distributed system performance analysis. Mascots; Published by the IEEE Computer Society; 26-2000.
23. Chen MY, Kiciman E, Fratkin E, Fox A, Brewer E. Pinpoint: Problem determination in large, dynamic internet services. Dependable systems and networks, 2002. DSN 2002. proceedings. international conference on; ; 27-2002.
24. Jiang G, Chen H, Yoshihira K, Saxena A. Ranking the importance of alerts for problem determination in large computer systems. Proceedings of the 6th international conference on autonomic computing; ACM; 38-2009.
25. Pertet S, Gandhi R, Narasimhan P. Fingerpointing correlated failures in replicated systems. Proceedings of the second workshop on tackling computer systems problems with machine learning; ; 19-2007.
26. Cherkasova L, Ozonat K, Mi N, Symons J, Smirni E. Automated anomaly detection and performance modeling of enterprise applications. 09. 09-2009;27(3):1-32.
27. Sandeep SR, Swapna M, Niranjan T, Susarla S, Nandi S. CLUEBOX: A performance log analyzer for automated troubleshooting. 18. 18-2008.
28. Lu Y, Cohen I, Zhou XS, Tian Q. Feature selection using principal feature analysis. Proceedings of the 15th international conference on multimedia; ACM; 2007.
29. Breiman L. Random forests. Mach Learning. 2001;45(1):5-32.
30. Kaufman L, Rousseeuw PJ. Finding groups in data: An introduction to cluster analysis. Wiley Online Library; 1990.
31. Hoke E, Sun J, Strunk JD, Ganger GR, Faloutsos C. Intemon: Continuous mining of sensor data in large-scale self-infrastructures. 23. 23-2006;40(3):38-44.
32. Hoke E, Sun J, Faloutsos C. Intemon: Intelligent system monitoring on large clusters. Proceedings of the 32nd international conference on very large data bases; VLDB Endowment; 22-2006.
33. Papadimitriou S, Sun J, Faloutsos C. Streaming pattern discovery in multiple time-series. Proceedings of the 31st international conference on very large data bases; VLDB Endowment; 2005.
34. Heo J, Henriksson D, Liu X, Abdelzaher T. Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study. Real-time systems symposium, 2007. RTSS 2007. 28th IEEE international; IEEE; 2007.
35. Elnozahy M, Kistler M, Rajamony R. Energy conservation policies for web servers. Proceedings of the 4th USENIX symposium on internet technologies and systems; Citeseer; 2003.
36. Bianchini R, Rajamony R. Power and energy management for server systems. Computer. 2004;37(11):68-76.
37. Khan MMH, Heo J, Li S, Abdelzaher T. Understanding vicious cycles in server clusters. Distributed computing systems (ICDCS), 2011 31st international conference on; IEEE; 70-2011.
38. Jiang ZM, Hassan AE, Hamann G, Flora P. An automated approach for abstracting execution logs to execution events. 43. 43-2008;20(4):249-67.
39. Xu W, Huang L, Fox A, Patterson D, Jordan M. Mining console logs for large-scale system problem detection. Proceedings of the third conference on tackling computer systems problems with machine learning techniques; USENIX Association; 07-2008.
40. Kutare M, Eisenhauer G, Wang C, Schwan K, Talwar V, Wolf M. Monalytics: Online monitoring and analytics for managing large scale data centers. Proceeding of the 7th international conference on autonomic computing; ACM; 48-2010.
41. Xu W, Huang L, Fox A, Patterson D, Jordan M. Experience mining google's production console logs. Workshop on managing systems via log analysis and machine learning technqiues (SLAML 2010); October 3rd; Vancomer, British Columbia. ; 03-2010.
42. Xu W, Huang L, Fox A, Patterson D, Jordan MI. Detecting large-scale system problems by mining console logs. Proceedings of the ACM SIGOPS 22nd symposium on operating systems principles; ACM; 04-2009.
43. Xu W, Huang L, Fox A, Patterson D, Jordan M. Online system problem detection by mining patterns of console logs. Data mining, 2009. ICDM'09. ninth IEEE international conference on; IEEE; 06-2009.
44. Xu W, Huang L, Fox A, Patterson D, Jordan M. Largescale system problems detection by mining console logs. 05-2009. Report No.: UCB/EECS-2009-103.
45. Bodík P, Goldszmidt M, Fox A, Andersen H. Fingerprinting the datacenter: Automated classification of performance crises. 08. 08-2009.
46. Miao Jiang, Munawar MA, Reidemeister T, Ward PAS. Automatic fault detection and diagnosis in complex software systems by information-theoretic monitoring. Dependable systems & networks, 2009. DSN '09. IEEE/IFIP international conference on; ; 11-2009.
47. Cohen I, Goldszmidt M, Kelly T, Symons J, Chase JS. Correlating instrumentation data to system states: A building block for automated diagnosis and control. OSDI'04: Proceedings of the 6th conference on symposium on opearting systems design \& implementation; San Francisco, CA. Berkeley, CA, USA: USENIX Association; 24-2004.
48. Gunasekaran R, Dillow DA, Shipman GM, Maxwell DE, Hill JJ, Park BH, et al. Correlating log messages for system diagnostics. Cray users group conference; 01 May; Edinburgh, United Kingdom. Oak Ridge National Laboratory (ORNL); Center for Computational Sciences; 01-2010.